This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for communicating with a device, comprising:

executing a kernel module in a memory;

executing at least one kernel thread in the memory to execute device driver functions for the kernel module, wherein the device driver functions are capable of being invoked by system calls from applications operating in a user context; [[and]]

executing, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context;

buffering a parameter list;

accessing, with one of the at least one kernel thread, device information from the device;

setting device values in the buffered parameter list to the accessed device information from the device;

setting a flag indicating that the at least one kernel thread needs to set parameters at the device to d evice parameter values set in the buffered parameter list; and

processing, by one of the at least one kernel thread, the buffered parameter list in response to the flag being set by performing:

applying a lock on information in the buffered parameter list including the device parameter values;

after applying the lock, copying the device parameter values from the buffered parameter list to a temporary buffer, wherein parameters at the device are set to the device parameter values from the buffered parameter list in the temporary buffer; and

releasing the lock after copying the device parameter values from the buffered parameter list to the temporary buffer.

2. (Original) The method of claim 1, wherein the kernel module spawns the at least one kernel thread to execute the calls to the device driver functions for the kernel module.

3. (Canceled)

4. (Previously Presented) The method of claim 1, wherein a kernel module function requests device information, further comprising:

in response to the request for the device information, accessing the buffered device information.

5. (Previously Presented) The method of claim 1, wherein the at least one kernel thread accesses buffered device information periodically and independently of kernel module requests for the device information.

6. (Canceled)

7. (Canceled)

8. (Previously Presented) The method of claim 1, further comprising:

spawning a kernel thread to set device parameters to parameter values buffered in the parameter list.

9. (Previously Presented) The method of claim 8, wherein the kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

10. (Canceled)

11. (Previously Presented) The method of claim 1, further comprising:

disabling higher priority contexts before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

12. (Original) The method of claim 11, wherein the higher priority context comprises a bottom half or Interrupt Request (IRQ) context.

13. (Previously Presented) The method of claim 1, further comprising:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

14. (Original) The method of claim 1, further comprising:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.

15. (Currently Amended) A system, comprising:

a network device;

a memory;

a processor executing code to perform:

execute a network device driver in memory to control the network device;

execute a kernel module in the memory;

execute at least one kernel thread in the memory to execute device driver functions for the kernel module, wherein the device driver functions are capable of being invoked by system calls from applications operating in a user context; [[and]]

execute, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context;

buffering a parameter list;

accessing, with one of the at least one kernel thread, device information from the device;

setting device values in the buffered parameter list to the accessed device information from the device;

setting a flag indicating that the kernel thread needs to set parameters at the device to device parameter values set in the buffered parameter list; and

processing, by one of the at least one kernel thread, the buffered parameter list in response to the flag being set by performing:

applying a lock on information in the buffered parameter list including device parameter values;

after applying the lock, copying the device parameter values from the buffered parameter list to a temporary buffer, wherein parameters at the device are set to the parameter values from the buffered parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the buffered parameter list to the temporary buffer.

16. (Previously Presented) The system of claim 15, wherein the kernel module spawns the at least one kernel thread to execute the called device driver functions.

17. (Canceled)

18. (Previously Presented) The system of claim 15, wherein a kernel module function requests device information, wherein the processor executes the code to further perform:

in response to the request for the device information, accessing the buffered device information.

19. (Currently Amended) The system of claim 15, wherein the [[the]] at least one kernel thread accesses device information periodically and independently of kernel module requests for device information.

20. (Canceled)

21. (Canceled)

22. (Previously Presented) The system of claim 15, wherein the at least one kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

23. (Canceled)

24. (Previously Presented) The system of claim 15, wherein the processor executes the code to further perform:

disabling higher priority context before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

25. (Previously Presented) The system of claim 15, wherein the processor executes the code to further perform:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

26. (Original) The system of claim 15, wherein the processor executes the code to further perform:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.

27. (Currently Amended) An article of manufacture comprising a computer readable storage medium having code executed by a processor for communicating with a device and to perform operations, the operations comprising:

executing a kernel module;

executing at least one kernel thread to execute device driver functions for the kernel module, wherein the device driver functions are capable of being invoked system calls from applications operating in a user context;

executing, with the at least one kernel thread, calls to device driver functions for the kernel module running in a kernel context;

buffering a parameter list;

accessing, with one of the at least one kernel thread, device information from the device;

setting device values in the buffered parameter list to the accessed device information from the device;

setting a flag indicating that the at least one kernel thread needs to set parameters at the device to device parameter values set in the buffered parameter list; and

processing, by one of the at least one kernel thread, ~~processes~~ the buffered parameter list in response to the flag being set by performing:

applying a lock on information in the buffered parameter list including the device parameter values;

after applying the lock, copying the device parameter values from the buffered parameter list to a temporary buffer, wherein parameters at the device are set to the device parameter values from the buffered parameter list in the temporary buffer; and

releasing the lock after copying the parameter values from the buffered parameter list to the temporary buffer.

28. (Original) The article of manufacture of claim 27, wherein the kernel module spawns at least one kernel thread to execute the called device driver functions.

29. (Canceled)

30. (Previously Presented) The article of manufacture of claim 27, wherein a kernel module function requests device information, wherein the operations further comprise:

in response to a request for the device information, accessing the buffered device information.

31. (Previously Presented) The article of manufacture of claim 27, wherein the at least one kernel thread accesses buffered device information periodically and independently of kernel module requests for device information.

32. (Canceled)

33. (Canceled)

34. (Previously Presented) The article of manufacture of claim 27, wherein the at least one kernel thread spawned to set device parameter values processes the parameter list to locate buffered parameter values and set the device parameters to the buffered parameter values.

35. (Canceled)

36. (Previously Presented) The article of manufacture of claim 27, wherein the operations further comprise:

disabling higher priority contexts before locking the parameter list; and

enabling the higher priority contexts after releasing the lock on the parameter list.

37. (Original) The article of manufacture of claim 36, wherein the higher priority context comprises a bottom half or Interrupt Request (IRQ) context.

38. (Previously Presented) The article of manufacture of claim 27, wherein the operations further comprise:

after releasing the lock, executing device driver functions to configure the device with the parameter values in the temporary buffer.

39. (Original) The article of manufacture of claim 27, wherein the code executes operations to further perform:

initiating, with the kernel module, an access request with respect to device information;

disabling any higher priority contexts capable of accessing the device information;

obtaining a lock for the device information subject to the access request;

providing the kernel module access to the device information;

releasing the lock; and

enabling all higher priority contexts that were disabled.